CSE440: Introduction to HCI
Methods for Design, Prototyping and Evaluating User Interaction

Lecture 14: Patterns

Nigini Oliveira
Abhinav Yadav
Liang He
Angel Vuong
Jeremy Viny
<table>
<thead>
<tr>
<th>May 13</th>
<th>May 14</th>
<th>May 15</th>
<th>May 16</th>
<th>May 17</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sb - Paper Prototype</td>
<td>Accessibility [slides]</td>
<td>Ngini's office hours</td>
<td>Patterns</td>
<td>Section</td>
</tr>
<tr>
<td>10:00 - 11:20</td>
<td>OUG 136</td>
<td>10:00 - 12:00</td>
<td>10:00 - 11:20</td>
<td>OUG 136</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allen Center 338</td>
<td>3b - Heuristic Evaluation</td>
<td>10:30 - 11:20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11:30 - 12:20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1:30 - 2:20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2:30 - 3:20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reading 3: Usability Testing</td>
<td>3c - Usability Testing Check-In (for Saturday)</td>
</tr>
<tr>
<td>May 20</td>
<td>May 21</td>
<td>May 22</td>
<td>May 23</td>
<td>May 24</td>
</tr>
<tr>
<td>Interface Implementation</td>
<td>3d - Usability Testing Review</td>
<td>Recap and Exam Q&amp;A Session</td>
<td>Section</td>
<td></td>
</tr>
<tr>
<td>10:00 - 11:20</td>
<td>OUG 136</td>
<td>Ngini's office hours</td>
<td>10:00 - 11:20</td>
<td>OUG 136</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10:00 - 12:00</td>
<td></td>
<td>MGH 058</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allen Center 338</td>
<td></td>
<td>MGH 058</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MGH 058</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MGH 058</td>
</tr>
<tr>
<td>May 27</td>
<td>May 28</td>
<td>May 29</td>
<td>May 30</td>
<td>May 31</td>
</tr>
<tr>
<td>Sb - Digital Mockup</td>
<td>3f - Workshop</td>
<td>Ngini's office hours</td>
<td>Exam</td>
<td>Section</td>
</tr>
<tr>
<td>10:00 - 11:20</td>
<td>OUG 136</td>
<td>10:00 - 12:00</td>
<td>10:00 - 11:20</td>
<td>OUG 136</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allen Center 338</td>
<td></td>
<td>MGH 058</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MGH 058</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MGH 058</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3f - Getting the Design Right</td>
<td>10:30 - 11:20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11:30 - 12:20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1:30 - 2:20</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2:30 - 3:20</td>
</tr>
</tbody>
</table>
Patterns

“… a general repeatable solution to a commonly-occurring usability problem in interface design…”

Patterns
Site-branding
What can I do here?
What if I choose a CD to buy?
Consistency to reinforce branding
User location within a site

Electronic commerce platform showing a product listing for Weezer (2001) by Weezer. The product is available for purchase at a best price of $6.99. The page includes options to view additional information, add to wish list, and buy the product. The platform also highlights various features such as Bestsellers, Music, Books, and more.
Clicked BUY:
Do we still know where we are?
Do we know what can we do?
Where am I in the process?
Where am I in the process?

Step 1 - Choose Shipping Address

Ship my order to:

Jason Hong
387 Soda Hall Computer Science UC Berkeley
Berkeley, CA 94720

Use This Address

OR

Enter a new shipping address:

Name
Street address
City
State
ZIP code
Country

Save Changes
Noticed anything different?
Process funnel
Extraneous info and links are removed to focus customers
Why this completed outdated example?
Why this completed outdated example?

It’s still the same today :)

Activity

5min

Go to your favorite shopping site and find the differences!
Design Equals Solutions
Design Equals Solutions

Designers often reinvent

- Hard to know how things were done before
- Why things were done a certain way
- How to reuse solutions
Design Patterns

Design patterns communicate common design problems and solutions

A tool for knowledge sharing
Design Patterns in Architecture

Design patterns communicate common design problems and solutions.
Design Patterns in OOP and software engineering

Nice patterns for creating reusable OO software.

The Gang of Four Book!
Using Design Patterns

Not too general and not too specific

use a solution “a million times over, without ever doing it the same way twice”
Using Design Patterns

Not too general and not too specific

   use a solution “a million times over, without ever doing it the same way twice”

Design patterns are a language

   For sharing the key elements of a solution
      Beer hall is part of a center for public life
      Beer hall needs spaces for groups to be alone ALCOVES
Web Design Patterns

Communicate design problems & solutions

how to create navigation bars for finding relevant content

how to create a shopping cart that supports check out

how to make e-commerce sites where people return & buy
Navigation Bar (K2)

Problem: Customers need a structured, organized way of finding the most important parts of your Web site
Hamburger Menu
Patterns Support Creativity

Patterns come from successful examples
   So successful that lots of people are familiar with
     their paradigms, interaction, techniques, and metaphors
     makes it work well across many sites (e.g., shopping carts)

Not too general and not too specific
   You need to specialize to your needs

Patterns let you focus on the hard, unique problems of your
c design situation
Principles, Guidelines, Templates

Principles?
- patterns are not that general

Guidelines?
- patterns discuss trade-offs,
- show good examples, and tie to other patterns

Style guides?
- patterns are not too specific,
- can be specialized to a design

Templates?
- patterns illustrate flows and relationships among pages

Patterns help design without over-constraining
UI Pattern Form

**Problem**: Problems are related to the usage of the system and are relevant to the user or any other stakeholder that is interested in usability.

**Use when**: a situation (in terms of the tasks, the users and the context of use) giving rise to a usability problem.

**Principle**: a pattern is usually based on one or more ergonomic principles such as user guidance, or consistency, or error management.

**Solution**: a proven solution to the problem.

**Why**: How and why the pattern actually works, including an analysis of how it may affect certain attributes of usability.

**Examples**: Each example shows how the pattern has been successfully applied in a real life system. This is often accompanied by a screenshot and a short description.

**Implementation**: Some patterns provide implementation details.

---


Example of a UI pattern

Undo in Microsoft Word using a visualization of the command queue

Example of a UI pattern

Author: Martijn van Welie  
http://www.welie.com/patterns/gui/undo.html

Problem: Users do actions they later want reverse because they realized they made a mistake or because they changed their mind.
Example of a UI pattern

Use when

You are designing a desktop or web-based application where users can manage information or create new artifacts. Typically, such systems include editors, financial systems, graphical drawing packages, or development environments. Such systems deal mostly with their own data and produce only few non-reversible side-effects, like sending of an email within an email application. Undo is not suitable for systems where the majority of actions is not reversible, for example, workflow management systems or transaction systems in general.

Both novice and expert users may want to reverse their actions, either because of mistakes or changes in intention. Expert users may want to use the history of their actions for more specific manipulation of the data in the application. For example, in a graphical modeling application, users may want to undo work on some specific object while keeping later work done on other objects.
Example of a UI pattern

Solution

Maintain a list of user actions and allow users to reverse selected actions.

Each 'action' the user does is recorded and added to a list. This list then becomes the 'history of user actions' and users can reverse actions from the last done action to the first one recorded. This is also called a Linear Multi-level Undo.
Example of a UI pattern

**Why**

Offering the possibility to always undo actions gives users a comforting feeling. It helps the users feel that *they* are in control of the interaction rather than the other way around. They can explore, make mistakes and easily go some steps back, which facilitates learning the application's functionality. It also often eliminates the need for annoying warning messages since most actions will not be permanent.
Example of a UI pattern

As in all MS Office applications, in Word 2000 the users can see the history of their actions and undo one or more of them. The actions are briefly described and the users can select a range of actions to be undone. After selecting undo, users can even redo the actions.

In Photoshop a selective undo is also possible. By moving the slider, users can do the normal multi-level undo but they can also drag an action into the trashcan and thereby do a selective undo. Operations that depended on that action are automatically deleted as well if they are not relevant anymore.
Example of a UI pattern

Implementation

Most implementations of multi-level undo are based on the Command (Gamma et al 1995) pattern. When using the Command pattern, most functionality is encapsulated in Command objects rather than in other controlling classes. The idea is to have a base class that defines a method to "do" a command, and another method to "undo" a command. Then, for each command, you derive from the command base class and fill in the code for the do and undo methods. The "do" method is expected to store any information needed to "undo" the command. For example, the command to delete an item would remember the content of the item being deleted.
More examples

Captcha

Problem
The application needs confirmation that the action that is about to happen is done by a human rather than an automated machine.

Solution
Present users with a mangled image containing numbers and letters that humans can still 'decipher' but is hard for machines to read.

Verify Your Registration

Enter the code shown: [Image with code]

This helps Yahoo! prevent automated registrations.

http://www.welie.com
More examples

Date Selector

Problem

Users need to select a date or a time period

Solution

Use a combination of an edit box and a graphical clickable calendar

From www.expedia.com

http://www.welie.com
Pre-patterns

Patterns require broad adoption and examples
- Many version of the same basic idea
- Shown successful in many contexts
- That is what makes them patterns

This is challenging in novel domains

Pre-patterns are based in weaker evidence
- Can help speed diffusion of techniques and results
- Can help see relationships among ideas
Dark Patterns

A design that has been carefully crafted to trick people into doing things, such as buying insurance with their purchase or signing up for recurring bills.
Dark Patterns

A design that has been carefully crafted to trick people into doing things, such as buying insurance with their purchase or signing up for recurring bills.

Example: Disguised Ads

Ads that are disguised as other kinds of content or navigation, in order to get users to click on them.
Dark Patterns

A design that has been carefully crafted to trick people into doing things, such as buying insurance with their purchase or signing up for recurring bills.

Example: Disguised Ads
- Ads that are disguised as other kinds of content or navigation, in order to get users to click on them

Example: Friend Spam
- A site or game asks for your credentials, then goes on to publish content or send out bulk messages
Dark Patterns

After Lawsuit Settlement, LinkedIn’s Dishonest Design Is Now A $13 Million Problem

Hopefully, this will be a lesson to other companies who use dark UX patterns to trick their users.

Anyone who has ever signed up, or even known anyone who has signed up, for LinkedIn has probably found themselves on the receiving end of dozens of follow-up emails, inviting you to “expand your professional network.” Even worse, they’re virtually impossible to opt-out of. It’s a scummy use of dark UX patterns by a company that should know better. Now, LinkedIn is going to be paying for it as part of a class-action lawsuit, to the tune of $13 million.

Presented in San Jose’s U.S. District Court, the key issue in Perkins v. LinkedIn is spam. Namely, during the user sign-up process, LinkedIn claims that it “will not store your password or email anyone without your permission.” Despite this, LinkedIn sends automated follow-up email reminders on a new user’s behalf to any contacts harvested from their webmail accounts, which are presented in such a way as to appear as if they came directly from the user.

Under California law, the sitting judge says has deemed this illegal. Consequently, if you were a member of LinkedIn’s ‘add connection’ program between September 2011 and October 2014, you can submit a claim to get a payout.
Pattern exercise

In preparation for the exam:

Read and answer the handout.
Ask me something!